ACCELERATED GRAPHICS PORT FOR A MULTIPLE MEMORY CONTROLLER COMPUTER SYSTEM

Cross Reference to Related Applications

[0001] This patent application is a continuation of and incorporates by reference, in its entirety, U.S. Patent Application No. 09/892, 917, filed June 26, 2001, which is a continuation o f U.S. Patent Application No. 09/000,511 filed on December 30, 1997, now Patent No. 6,252,612. The patents and patent applications listed below are related to the present application, and are each hereby incorporated by reference in their entirety.

SYSTEM FOR ACCELERATED GRAPHICS PORT ADDRESS REMAPPING INTERFACE TO MAIN MEMORY, U.S. Patent No. 6,073,198, filed on June 25, 1997; Issued on May 30, 2000.

METHOD OF IMPELEMENTING AN ACCELERATED GRAPHICS PORT FOR A MULTIPLE MEMORY CONTROLLER COMPUTER SYSTEM, U.S. Patent No. 6,157,398, filed on December 30, 1997; Issued on December 5, 2000

APPARATUS FOR GRAPHIC ADDRESS REMAPPING, U.S. Patent No. 6,249,853, filed on June 25, 1997; Issued on June 19, 2001.

METHOD FOR PERFORMING GRAPHIC ADDRESS REMAPPING, U.S. Patent No. 6,282,625, filed on June 25, 1997; issued August 28, 2001.

METHOD OF IMPELEMENTING AN ACCELERATED GRAPHICS PORT FOR A MULTIPLE MEMORY CONTROLLER COMPUTER SYSTEM, U.S. Patent Application No. 09/723,403, filed on November 27, 2000.

Background of the Invention

Field of the Invention

[0002] The present invention relates to computer systems, and more particularly, to a method of using a second memory controller having an accelerated graphics port.

Description of the Related Technology

[0003] As shown in Figure 1, a conventional computer system architecture 100 includes a processor 102, system logic 104, main memory 106, a system bus 108, a graphics accelerator 110 communicating with a local frame buffer 112 and a plurality of peripherals 114. The processor 102 communicates with main memory 106 through a memory management unit (MMU) in the processor 102. Peripherals 114 and the graphics accelerator 110 communicate with main memory 106 and system logic 104 through the system bus 108. The standard system bus 108 is currently the Peripherals Component Interface (PCI). The original personal computer bus, the Industry Standard Architecture (ISA), is capable of a peak data transfer rate of 8 megabytes/sec and is still used for low-bandwidth peripherals, such as audio. On the other hand, PCI supports multiple peripheral components and add-in cards at a peak bandwidth of 132 megabytes/sec. Thus, PCI is capable of supporting full motion video playback at 30 frames/sec, true color high-resolution graphics and 100 megabits/sec Ethernet local area networks. However, the emergence of high-bandwidth applications, such as three dimensional (3D) graphics applications, threatens to overload the PCI bus.

[0004] For example, a 3D graphics image is formed by taking a two dimensional image and applying, or mapping, it as a surface onto a 3D object. The major kinds of maps include texture maps, which deal with colors and textures, bump maps, which deal with physical surfaces, reflection maps, refraction maps and chrome maps. Moreover, to add realism to a scene, 3D graphics accelerators often employ a z-buffer for hidden line removal and for depth queuing, wherein an intensity value is used to modify the brightness of a pixel as a function of distance. A z-buffer memory can be as large or larger than the memory needed to store two dimensional images. The graphics accelerator 110 retrieves and manipulates image data from the local frame buffer 112, which is a type of expensive high performance memory. For example, to transfer an average 3D scene (polygon overlap of three) in 16-bit color at 30 frames/sec at 75 Hz screen refresh, estimated bandwidths of 370 megabytes/sec to 840 megabytes/sec are needed for screen resolutions from 640 x 480 resolution (VGA) to 1024 x 768 resolution (XGA). Thus, rendering of 3D graphics on a

display requires a large amount of bandwidth between the graphics accelerator 110 and the local frame buffer 112, where 3D texture maps and z-buffer data typically reside.

[0005] In addition, many computer systems use virtual memory systems to permit the processor 102 to address more memory than is physically present in the main memory 106. A virtual memory system allows addressing of very large amounts of memory as though all of that memory were a part of the main memory of the computer system. A virtual memory system allows this even though actual main memory may consist of some substantially lesser amount of storage space than is addressable. For example, main memory may include sixteen megabytes (16,777,216 bytes) of random access memory while a virtual memory addressing system permits the addressing of four gigabytes (4,294,967,296 bytes) of memory.

[0006] Virtual memory systems provide this capability using a memory management unit (MMU) to translate virtual memory addresses into their corresponding physical memory addresses, where the desired information actually resides. A particular physical address holding desired information may reside in main memory or in mass storage, such as a tape drive or hard disk. If the physical address of the information is in main memory, the information is readily accessed and utilized. Otherwise, the information referenced by the physical address is in mass storage and the system transfers this information (usually in a block referred to as a page) to main memory for subsequent use. This transfer may require the swapping of other information out of main memory into mass storage in order to make room for the new information. If so, the MMU controls the swapping of information to mass storage.

[0007] Pages are the usual mechanism used for addressing information in a virtual memory system. Pages are numbered, and both physical and virtual addresses often include a page number and an offset into the page. Moreover, the physical offset and the virtual offset are typically the same. In order to translate between the virtual and physical addresses, a basic virtual memory system creates a series of lookup tables, called page tables, stored in main memory. These page tables store the virtual address page numbers used by the computer. Stored with each virtual address page number is the corresponding physical

address page number which must be accessed to obtain the information. Often, the page tables are so large that they are paged themselves. The page number of any virtual address presented to the memory management unit is compared to the values stored in these tables in order to find a matching virtual address page number for use in retrieving the corresponding physical address page number.

[0008] There are often several levels of tables, and the comparison uses a substantial amount of system clock time. For example, to retrieve a physical page address using lookup tables stored in main memory, the typical MMU first looks to a register for the address of a base table which stores pointers to other levels of tables. The MMU retrieves this pointer from the base table and places it in another register. The MMU then uses this pointer to go to the next level of table. This process continues until the physical page address of the information sought is recovered. When the physical address is recovered, it is combined with the offset furnished as a part of the virtual address and the processor uses the result to access the particular information desired. Completion of a typical lookup in the page tables may take from ten to fifteen clock cycles at each level of the search. Such performance is unacceptable in processing graphical applications.

[0009] One solution to facilitate the processing of graphical data includes having a point to point connection between the memory controller and a graphics accelerator. Such an architecture is defined by the *Accelerated Graphics Port Interface Specification, Revision 1.0*, (July 31, 1996) released by Intel Corporation. However, one problem with these systems is that the PCI bus acts as a bottleneck for all memory transactions. Computer manufacturers are in need of a system to eliminate this bottleneck.

[0010] Other solutions to facilitate the access of memory exist. The U.S. Patent No. 4,016,545 to Lipovski teaches the use of multiple memory controllers. However, Lipovski does not describe a point to point connection between a memory controller and a graphics accelerator. Such a connection is needed for the high speed processing of graphic data.

[0011] Additionally, U.S. Application No. 4,507,730 to Johnson teaches the use of multiple memory controllers. However, Johnson uses multiple memory controllers for

fault tolerance. In Johnson, once a memory controller is found to be faulty, it is switched off line and another memory controller is activated in its place. The memory controllers in Johnson do not facilitate the efficient transfer of memory for graphic applications.

[0012] In view of the limitations discussed above, computer manufacturers require an architecture with improved methods for storing, addressing and retrieving graphics data from main memory. Moreover, to address the needs of high bandwidth graphics applications without substantial increases in system cost, computer manufacturers require improved technology to overcome current system bus bandwidth limitations.

Summary of the Invention

[0013] One embodiment of the invention is a method of manufacturing a multiple memory controller computer comprising connecting at least two memory controllers to at least one processing unit; and connecting at least one configuration register to one of the at least two memory controllers, wherein the at least one configuration register defines a range of addresses that are available for accelerated graphic port transactions.

[0014] Yet another embodiment of the invention is a method of using a multiple memory controller system, comprising storing a graphical address remapping table in a memory on a computer system having at least two memory controllers; connecting a graphics accelerator to a memory controller which has at least one configuration register that defines a range of addresses that are available for accelerated graphics port transactions; and storing a graphics address relocation table in a memory connected to said memory controller having at least one configuration register.

Brief Description of the Drawings

[0015] Figure 1 is a block diagram illustrating the architecture of a prior art computer system.

[0016] Figure 2 is a block diagram illustrating one embodiment of a computer system of the invention.

[0017] Figure 3 is a block diagram illustrating the address space of a processor of one embodiment of the invention.

[0018] Figure 4 is a block diagram illustrating a second embodiment of the invention.

[0019] Figure 5 is a block diagram illustrating the translation of a virtual address to a physical address of one embodiment of the invention.

[0020] Figure 6 is an illustration of a page table entry of the graphics address remapping table of one embodiment of the invention.

[0021] Figure 7 is a block diagram illustrating the generation of a translation lookaside buffer entry of one embodiment of the invention.

Detailed Description of the Invention

[0022] The following detailed description presents a description of certain specific embodiments of the invention. However, the invention can be embodied in a multitude of different ways as defined and covered by the claims. In this description, reference is made to the drawings wherein like parts are designated with like numerals throughout.

[0023] Figure 2 is a block diagram illustrating a computer system of one embodiment of the invention. This computer 150 includes at least one processor 152 connected to a first memory controller 154 and a second memory controller 155 by a processor or host bus. The computer 150 also has a first main memory 156 and a second main memory 157 connected to the first memory controller 154 and the second memory controller 155, respectively. A graphics accelerator 160 communicates with a local frame buffer 162 and the first memory controller 154 through an accelerated graphics port (AGP) 166. The AGP-166 is not a bus, but is a point-to-point connection between an AGP compliant target which is the first memory controller 154, and an AGP-compliant master, which is the graphics accelerator 160. The AGP 166 point-to-point connection enables data transfer on both the rising and falling clock edges, improves data integrity, simplifies AGP protocols and eliminates bus arbitration overhead. AGP provides a protocol enhancement enabling pipelining for read and write accesses to the main memory 156. The first memory

controller 154 and the second memory controller 155 also accept memory requests from a PCI bus 158.

[0024] As noted above, the embodiment of Figure 2 enables the graphics accelerator 160 to access both the first main memory 156 and the local frame buffer 162. From the perspective of the graphics accelerator 160, the main memory 156 and the local frame buffer 162 are logically equivalent. Thus, to optimize system performance, graphics data may be stored in either the first main memory 156 or the local frame buffer 162. In contrast to the direct memory access (DMA) model where graphics data is copied from the main memory 156 into the local frame buffer 162 by a long sequential block transfer prior to use, the graphics accelerator 160 of the present invention can also use, or "execute," graphics data directly from the memory in which it resides (the "execute" model).

[0025] The interface between the first memory controller 154 and the graphics accelerator 160 is defined by *Accelerated Graphics Port Interface Specification, Revision 1.0*, (July 31, 1996) released by Intel Corporation and available from Intel in Adobe® Acrobat® format on the World Wide Web at the URL: *developer.intel.com/pc-supp/platform/agfxport/INDEX.htm*. This document is hereby incorporated by reference.

[0026] Figure 3 illustrates an embodiment of the address space 180 of the computer system 150 (Figure 2) of the invention. For example, a 32 bit processor 152 (Figure 2) has an address space 180 including 2^{32} (or 4,294,967,296) different addresses. A computer system 150 (Figure 2) typically uses different ranges of the address space 180 for different devices and system agents. In one embodiment, the address space 180 includes a graphics address remapping table (GART) range 184 and a main memory range 186.

[0027] The first memory controller 154 provides a set of registers to define the range of available for AGP transactions. A base register 165 is used to define the base address of the AGP addresses. A range register 166 is used to establish the amount of memory following the base address that is dedicated to AGP transactions. Alternatively, a lower and upper address register may be used to define the AGP address range. An operating system provided with these values will attempt to allocate GART pages within this memory

range. In contrast to prior art systems, the operating system attempts to first remap the addresses falling within the GART range 184 to the first memory controller 154.

[0028] By employing a first and second main memory 156, 157 respectively, and two memory controllers 154, 155 faster transaction processing is realized than in those prior art systems employing a single system memory and a single memory controller. In particular, two memory transactions can be executed simultaneously by executing one transaction using the first memory controller 154 while another transaction is being executed by the second memory controller 155. Graphics data typically is read many times without ever being changed or written to. Read and write delays are reduced by storing the graphic data in the first memory controller 154, while storing other data in the second memory controller 155.

[0029] Referring again to Figure 3, the computer 150 has 64 megabytes of main memory 218 encompassing physical addresses 0 through 0x03FFFFFF. 32 megabytes of memory are assigned to the first memory controller 154 and 32 megabytes are assigned to the second memory controller 155. Using the base 165 and range 166 registers provided by the first memory controller 154, the operating system has set the AGP related data occupying the lower 32 megabytes of the first main memory 156 referenced by physical addresses 0x00000000 through 0x01FFFFFF. For example, if the GART Range 184 begins at the 256 megabyte virtual address boundary 0x10000000, the invention enables translation of virtual addresses within the GART Range 184 to physical addresses in the lower 32 megabytes of the first main memory 156 corresponding to physical addresses in the range 0x00000000 through 0x01FFFFFF.

[0030] Upon a request from the graphics accelerator 160 the first memory controller 154 analyzes the address in the request to identify whether the address is in the first main memory 156. If the address is not within the first main memory 156, the first memory controller 154 re-routes the request to the second memory controller 155. By having the GART tables and their referenced memory located on the first memory controller 154 having the AGP, the re-routing of memory requests to the other memory controller 155 is minimized.

[0031] In one embodiment, a hardware abstraction layer (HAL) directs the operating system to place the GART table and texture memory in the first memory controller 154. The HAL is a small layer of software that presents the rest of the computer system with an abstract model of any hardware that is not part of the processors 152. The HAL hides platform-specific details from the rest of the system and removes the need to have different versions of the operating system for platforms from different vendors.

[0032] Referring to Figure 4, a second embodiment of the invention is illustrated. This second embodiment has a second memory controller 190 also having an accelerated graphics port 192 for use by a graphics accelerator 170. Each of the memory controllers 154, 190 provide a set of registers defining a range of addresses that are used by the operating system for accelerated graphics port transactions. In a third embodiment of the invention, a single chip contains a plurality of memory controllers each memory controller having an AGP and a set of configuration registers identifying a range of addresses that are used for AGP transactions.

[0033] Figure 5 illustrates the translation of a virtual address 200 to a physical address 202 in one embodiment of the invention. As discussed previously, in one embodiment, the operating system attempts to allocate those virtual addresses falling within the GART range 184 (Figure 3) to the first main memory 156 (Figure 3).

[0034] A virtual address 200 includes a virtual page number field 204 and an offset field 206. Translation of the contents of the virtual page number field 204 occurs by finding a page table entry (PTE) corresponding to the virtual page number field 204 among the plurality of GART PTEs 208 in the GART table 210. To identify the appropriate PTE having the physical address translation, the GART base address 212 is combined at a state 213 with the contents of the virtual page number field 204 to obtain a PTE address 214. The contents referenced by the PTE address 214 provide the physical page number 216 corresponding to the virtual page number 204. The physical page number 216 is then combined at a state 217 with the contents of the offset field 206 to form the physical address 202. The physical address 202 in turn references a location in the first main memory 156 having the desired information.

[0035] The GART table 210 may include a plurality of PTEs 208 having a size corresponding to the memory page size used by the processors 152 (Figure 2). For example, an Intel® Pentium® or Pentium® Pro processor operates on memory pages having a size of 4K. Thus, a GART table 210 adapted for use with these processors may include PTEs referencing 4K pages. In one embodiment, the virtual page number field 204 comprises the upper 20 bits and the offset field 206 comprises the lower 12 bits of a 32 bit virtual address 200. Thus, each page includes $2^{12} = 4096$ (4K) addresses and the lower 12 bits of the offset field 206 locate the desired information within a page referenced by the upper 20 bits of the virtual page number field 204.

[0036] Figure 6 illustrates one possible format for a GART PTE 220. The GART PTE 220 includes a feature bits field 222 and a physical page translation (PPT) field 224. In contrast to prior art systems where hardwired circuitry defines a page table format, the GART table 210 (Figure 5) may include PTEs of configurable length enabling optimization of table size and the use of feature bits defined by software. The PPT field 224 includes PPTSize bits to generate a physical address 202 (Figure 5). The PPTSize defines the number of translatable addresses.

[0037] In one embodiment, an initialization BIOS implements the GART table 210 (Figure 5) by loading configuration registers in the first memory controller 154 (Figure 2) during system boot up. In another embodiment, the operating system implements the GART table 210 (Figure 5) using an API to load the configuration registers in the first memory controller 154 (Figure 3) during system boot up.

[0038] As noted earlier, a GART table 210 includes multiple PTEs, each having physical page translation information 224 and software feature bits 222. The GART table 210 may be located at any physical address in the main memory 218, such as the 2 megabyte physical address 0x00200000. The operating system attempts to place the GART table 210 in the memory range provided by the registers 165, 166 in the first memory controller 154 if space is available. By placing the GART table 210 in this memory range, fewer memory requests from the graphic accelerator 160 need to travel over the PCI bus 158 to the second memory controller 166 as compared to traditional systems. For a system having a 4K

memory page size and a GART PTE 220 of 8 byte length, the GART table 210 is configured as follows:

PhysBase	:= 0x00000000	-- Start of remapped physical address
PhysSize	:= 32 megabytes	-- Size of remapped physical addresses
AGPAperture	:= 0x10000000	-- Start address of GART Range
GARTBase	:= 0x00200000	-- Start address of GART table
$2^{PTESize}$	:= 8 bytes	-- Size of each GART Page Table Entry
PageSize	:= 4 kilobytes	-- Memory page size

[0039] To determine the number of PTEs in the GART table 210, the size of the physical address space in main memory 218 allocated to AGP related data, the upper 32 megabytes = 33554432 bytes, is divided by the memory page size, 4K = 4096 bytes, to obtain 8192 PTEs. Since there are 8 bytes in each PTE, the GART table consists of 65,536 bytes (8192 x 8). Note that $8192 = 2^{13} = 2^{PTESize}$ and thus, PTESize = 13. Using the values supplied by the base and range registers, the operating system programs the configuration registers with the following values to set up the GART table 210:

PhysBase	:= 0x00000000	-- Start of remapped physical address
AGPAperture	:= 0x10000000	-- Start address of GART Range
GARTBase	:= 0x00000000	-- Start address of GART table
PTESize	:= 3	--$2^{PTESize}$ = Size in bytes of the PTE
PPTSize	:= 13	-- Number of PPT bits in each PTE

Base Register 165 := 0x00000000	--	Starting point of memory in the first memory controller 154
Range Register 166:= 0x01FFFFFF	--	Range of memory available for AGP transactions

[0040] Note that the operating system chose to set up the GARTBase and PhysBase in the range of addresses suggested by the base register 165 and range register 166 located in first memory controller 154.

[0041] Figure 7 illustrates the translation of a virtual address 200 to a physical address 202 (Figure 5a) using a translation lookaside buffer (TLB) 240. As before, a virtual address 200 includes a virtual page number field 204 and an offset field 206. Translation of the virtual page number field 204 occurs by finding a PTE of the GART table 210 corresponding to the contents of the virtual page number field 204. The GART base address

212 is combined at 213 with the contents of the virtual page number field 204 to obtain a PTE address 214. The PTE address 214 in turn provides the physical page number 216 corresponding to the virtual page number 204. At this point, a TLB entry 242 is formed having a virtual page field 246, its corresponding physical page field 244, a least recently used (LRU) counter 250 to determine the relative age of the TLB entry 242 and a status indicator 248 to determine when the TLB 240 has valid information. The TLB entry 242 is stored in a TLB 240 having a plurality of TLB entries 252. In one embodiment, there are a sufficient quantity of TLB entries 252 to cover all of the translatable addresses in the entire GART range 184 (Figure 3). In this embodiment, the first memory controller 154 (Figure 2) includes a block of registers to implement the TLB 240. In another embodiment, first memory controller 154 (Figure 2) includes a fast memory portion, such as cache SRAM, to implement the TLB 240.

[0042] The invention advantageously overcomes several limitations of existing technologies and alternatives. For example, the AGP connection can support data transfers over 500 megabytes a second. By defining a set of memory that is available for AGP transaction, operating systems can optimize system performance by keeping the graphic data on the memory controller with the accelerated graphics port. The memory controller having the accelerated graphics port handles memory transactions concurrently with transactions being processed by the other memory controller.

[0043] Additionally, the invention enables storing, addressing and retrieving graphics data from relatively inexpensive main memory without the bandwidth limitations of current system bus designs. It is to be noted that in an alternative embodiment of the invention, the memory controllers may be on the same semiconductor chip as the memory that they control.

[0044] In contrast to the conventional computer system architecture 100 (Figure 1), embodiments of the invention enable relocation of a portion of the 3D graphics data, such as the texture data, from the local frame buffer to main memory connected to a dedicated memory controller to reduce the size, and thus the cost, of the local frame buffer and to

improve system performance. For example, as texture data is generally read only, moving it to main memory does not cause coherency or data consistency problems.

[0045] Moreover, as the complexity and quality of 3D images has increased, leaving 3D graphics data in the local frame buffer 112 has served to increase the computer system cost over time. By moving 3D graphics data to a memory controller with its main memory, the architecture of the invention reduces the total system cost since it is less expensive to increase main memory 156 with a second controller 154 than to increase local frame buffer memory 112.

[0046] The invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiment is to be considered in all respects only as illustrative and not restrictive and the scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced with their scope.